Program System for Object Models Deductive Synthesis

Nataly Zhukova, Natalya Andrianova
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
St. Petersburg, Russia
nazhukova, natasha23062@mail.ru

NikolayKlimov
ITMO University
St. Petersburg, Russia
hocico16@gmail.com

***Abstract*—The paper considers the problem of dynamic modeling of complex natural and technical objects. The objects that have hierarchical structure are in focus. It is proposed to use new multilevel relatively finite automata models as formal models of such objects. A new algorithm based on deductive synthesis that allows automatically build automata models is presented. Automata models and the algorithm are implemented in program system. A number of examples of building models in the domain of Internet of Things are given.**

I. Introduction

The problem of modeling complex natural and technical objects in the dynamics begins to come to the fore.Machine models of real objects are in great demand in all spheres of human activity - social, economic and others. Such models are needed to solve many new problems. Thus, the specialists in the field of medicine justified the effectiveness of the application of a systematic approach to the analysis of patients' conditions [12]. This involves modeling the organism as an integral system in order to assess the consistency of the interaction of many subsystems of the organism. To do this, it is necessary to carry out many diagnostic measures, to identify the relationship between their results. In the field of education, an individual approach to learning is implemented, in accordance with which, an educational trajectory is built for each student taking into account its individual characteristics, existing competencies, and acquired skills [13], [14]. This requires a targeted search and linking of units of the educational information space within a single model. Simplier examples can be found in the field of the Internet of Things (IoT) [16]. IoT model, as a rule, contains three levels - sensing layer, relay layer, convergence layer. The state of IoT changes at each moment of time - some of the sensors fail, and new ones are added. The IoT model should allow assessing the state of the network as a whole and its readiness to solve applied problems. Many hundreds of such examples can be found in each area. The space of possible solutions of the considered class of problems turns out to be extremely large, the search for solutions in such a space requires large computational resources. From the IT side, a number of new technologies have been proposed for solving computationally complex problems, from computing using multiprocessor machines to cloud computing [15]. However, taking into account the observed dynamics of the development of modern society, the capabilities of the available technologies will soon be insufficient. We need new cost-effective solutions that will allow us to model complex objects in dynamics from the standpoint of system analysis.

The article proposes a new approach to modeling, based on the deductive synthesis of object models. In the second and third sections, the proposed models and methods of synthesis are considered. The fourth section describes the framework for deductive modeling. In the fifth section, examples of model synthesis for the field of IoT are considered.

II. Synthesis Basics

Dynamic models of an object are a sequence of interconnected transitions of the observed object between states. At each time, the object is in a state. Examples of such states are a healthy state, a state in the presence of an error. The state of an object at each moment of time is described by its model, which is built on facts known about the object. Such models are called static models. As a result of the binding of static models, a dynamic model is formed.

Automata models can be used to describe static and dynamic models of objects. To build models one can use known methods of program automatic synthesis [1], [2], [3], [4], [5].

When constructing static models, inductive synthesis algorithms are required. These algorithms and examples of their application are given in [6]. The binding of dynamic models is provided by deductive synthesis.When linking, a proof is made of the possibility of transition from one static model to another. If the proof was successful, a connection is established between the models. The proof is carried out at many levels determined by the structure of static models of objects. Connections established between models allow conclusions to be drawn about the dynamics and direction of changes occurring in an object.

An example of a dynamic model is shown in Fig. 1.

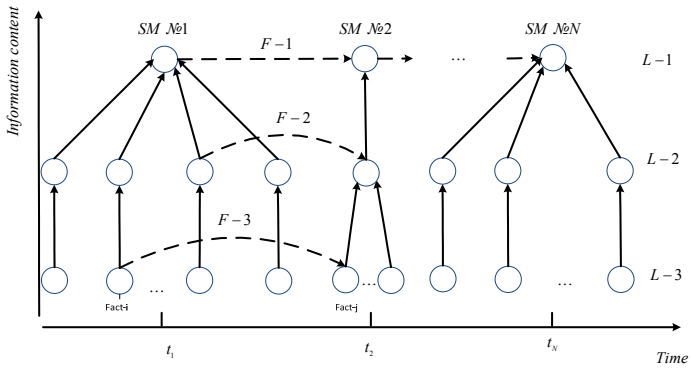

Fig. 1. Dynamic model

Fig. 1 shows N static models SM-1, SM-2, ..., SM-N, built at times t-1, t-2, t-N. All presented models have three levels. The elements of the model are some facts about the observed object. Each level has its own facts. The facts of adjacent levels of static models are interconnected. Facts related to the second level are determined on the basis of facts of the third level, and facts of the first level are determined on the basis of facts of the second level. The interrelationships between facts define vertical relationships in static models. For example, if sensors are located at the lower level that measure the values of parameters of the observed objects, then at the second level the total amount of data from sensors of a certain type can be calculated, and at the upper level - the total information content of all collected data.

In order to establish links between static models, first consider the elements of the top level. If such a link is established, then the corresponding horizontal top level link is established. In fig. 1 such a connection is designated as F-1. For example, when establishing a connection, such a criterion can be considered - the total amount of data decreased by no more than k%. If the connection fails, the transition to the second level is performed. At the second level, the amount of data on sensor types is investigated and F-2 type connections are established. If necessary, the transition to the third level is possible.

Automata models that allow describing static and dynamic models of observed objectsare considered below.

A. Automaton models of observed objects

Let us consider a formal model of observed objects. They can operate according to different programs. The states of any object with fixed structure can be described by the final state automate [7, 8]. For formal description of objects including self repairing and self reproductive objects it is necessary to take into account a number of additional conditions. These objects can be formalized by means of relative finite state operational automata.

Each automate RFA_r in r-th moment of time can be described in terms of 10 parameters,

$$RFA_r = \{\bar{d}_{a_r}, \bar{d}_{b_r}, \bar{d}_{c_r}, F_r^b, F_r^c, DA(\bar{d}_{b_{r-1}})\},$$
$$DB(\bar{d}_{b_{r-1}}), DC(\bar{d}_{b_{r-1}}), FB(\bar{d}_{b_{r-1}}), FC(\bar{d}_{b_{r-1}}), \quad (3)$$

where: $\bar{d}_{a_r}$ - input data vector; $\bar{d}_{b_r}$ - vector of internal state parameters; $\bar{d}_{c_r}$ - vector of output state parameters. Functions of transitions F_r^b in (3) define automata transitions from one internal state to another internal state,

$$\bar{d}_{b_{r+1}} = F_r^b(\bar{d}_{a_r}, \bar{d}_{b_r}) \quad (4)$$

Function of states of output F_r^c can be described as

$$\bar{d}_{c_r} = F_r^c(\bar{d}_{a_r}, \bar{d}_{b_r}) \quad (5)$$

States $\bar{d}_{b_r}$, $\bar{d}_{c_r}$, $\bar{d}_{a_r}$, and functions F_r^b, F_r^c, which define automate in r-th moment of time, must satisfy following conditions:

$$\bar{d}_{a_r} \in DA(\bar{d}_{b_{r-1}}) \quad (6)$$

$$\bar{d}_{b_r} \in DB(\bar{d}_{b_{r-1}}) \quad (7)$$

$$\bar{d}_{c_r} \in DC(\bar{d}_{b_{r-1}}) \quad (8)$$

$$F_r^b \in FB(\bar{d}_{b_{r-1}}) \quad (9)$$

$$F_r^c \in FC(\bar{d}_{b_{r-1}}) \quad (10)$$

Condition (6) says, that state of automate in r-th moment of time is limited by the set $DA(\bar{d}_{b_{r-1}})$ allowed states, defined for r-1 moment of time. According to (7) internal state of automate for r-th moment of time must be a member of the set $DB(\bar{d}_{b_{r-1}})$ of allowed internal states. Expression (8) defines limitations for allowed states of automate outputs. These states must be members of the set $DC(\bar{d}_{b_{r-1}})$. According to the condition (9) transition function F_r^b for r-th moment of time must be a member of the set $FB(\bar{d}_{b_{r-1}})$ of allowed functions for r-1 moment of time. The set $FB(\bar{d}_{b_{r-1}})$ of transition functions defines the instruction set of the automate for r-th moment of time. F_r^b is defined by the vector $\bar{d}_{b_r}$, which describes parameters of internal states of automate. According to (10) function of outputs F_r^c at r-th moment of time must be a member of the set $FC(\bar{d}_{b_{r-1}})$ of allowed functions, which are active at r-1 moment. Transition from automate RFA_r to automate RFA_{r+1} at r+1 moment of time one can describe as

$$F_r^b: RFA_r, \bar{d}_{a_r} \rightarrow RFA_{r+1}.$$

To build models of complex hierarchical objects that comply with real objects, multilevel relatively finite operational automata are required. The process of their construction has following steps: i) definition of the basic sets of allowed parameters of automata, ii) marking these sets by upper index «o», not taking into account their correlations with internal states. Let us define the complex of the basic sets as

$$DRFA^0 = \{DA^0, DB^0, DC^0, FB^0, FC^0\} \quad (11)$$

From elements of these basic sets (11) one can form

allowed sets of parameters of higher i-thlevels, $DRFA^i = \{DA^i, DB^i, DC^i, FB^i, FC^i\}$. As a result the automate may be characterized by allowed sets of parameters on different hierarchical levels,

$$DRFA^0 \Leftrightarrow DRFA^1 \Leftrightarrow \ldots \Leftrightarrow DRFA^i \Leftrightarrow \ldots \Leftrightarrow DRFA^K \quad (12)$$

Taking into account that automate for current moment of time is described by $DRFA^i = \{DA^i, DB^i, DC^i, FB^i, FC^i\}$ this complexes in general case are changing in time. They can be described as a function of its internal state

$$DRFA^i = DRFA^i(\bar{d}_{b_{r-1}}).$$

In a number of cases by means of expending of the set of automate internal states from (3) - (10) one can exclude functions (5) and correlated with them conditions (8), (10). As a result we receive automate reduced by parameters (4), (6), (7), (9), but with saving ability for reconfiguration.

$$RFA_r^* = \{\bar{d}_{a_r}, \bar{d}_{b_r}, F_r^b, DA(\bar{d}_{b_{r-1}}), DB(\bar{d}_{b_{r-1}}), FB(\bar{d}_{b_{r-1}})\}. \quad (13)$$

While using logical variant of presentation function of automate transition (4) has a view

$$F_r^b(\bar{d}_{a_r}, \bar{d}_{b_r}) \to \bar{d}_{b_{r+1}}. \quad (14)$$

If the set of internal states is expended not only by output states but also with input states then in (14) function $F_r^b(\cdot)$ from $\bar{d}_{a_r}$ maybe not shown.

Distinguishing feature of the described above relatively finite state operational automate is that the set of allowed parameters are true only on one stage (transition) and they are defined relatively to previous state. There is a possibility to change in a full automate not only the set of allowed input, output and internal states, but also the sets of transaction and output functions of automate. In particular cases full automate can be reduced to other automate, with allowed sets of parameters which do not depend upon previous internal states.

Automate (3) – (10) can be considered as a model of a fully reconfigurable object. Each such automate can be conceded as a complex of coupled automate of lower level. Migration between the levels from the formal point of view can be conceded as a process of tuning of the set of allowed parameters. The length of the record about the same functionality in the form of relatively finite state operational automate depends essentially upon the level of hierarchy used for automate operation description.

The distinguishing feature of multilevel models is that the synthesis of such models can be reduced to solving a small number $k \le K$ of simple problems. A low complexity of each problem is caused by a small number of analyzed conditions at each level.

Below the methods of multilevel dynamic model synthesis is described.

B. Synthesis algorithm of dynamic models

Automata models provide a description of the input and output data, conditions and functions of the automaton transition from one state to another. In the software implementation of the models, the high-level language (notation) JSON (JavaScript Object Notation) is used to describe therelative finite state operational automata.

Input and output data are a set of initial and resulting facts. They are specified as an array of arbitrary variables. For example, the input data can be: [a1, b1, c1], where a1, b1, c1 are some facts.

Transition conditions are specified as logical expressions and can be either static (initially set) or dynamic, i.e. redefined at each step of the synthesis.

The transition functions F from one state to another are specified as a JSON object of the following structure:

```
{
  "args": ["a1", "b1"],
  "conditions": "a1 < 10",
  "result ": "c2"
}
```

whereargs is the input to the function. They define the facts that are necessary to complete the transition; result - the result of the function. The result of the execution may be one or more new facts; conditions - conditions in the form of a logical expression that determine the possibility of using the function.

Simulation levels are defined as a hierarchical graph structure (“tree” or “forest”) in the following form:

```
[{"functions": [F1, F2],
"id": "baad85ac-1733-4f25-9609-e335807bbb4c",
"level": 1,
"number": 1,
"parent": null
}]
```

where functions is a set of transition functions defined at the level; id - the unique identifier of the element level; level - the ordinal number of the level; number - the sequence number of the item; parent is an optional identifier of the parent element from a higher level.

In multilevel modeling, the output facts proven at the child levels can be used as input facts when describing transition functions at their own level as well as at the parent levels.

Thus, the problem of synthesizing dynamic models is reduced to finding such functions F that allow one to prove the facts determined by the output data, in the presence of facts given as a set of input data. Let the initial facts be determined by a static model SM_{t_i} built at the time ti, and the output facts - by a static model $SM_{t_{i+k}}$, i.e. considered a sequence of k models.

The first step is to look at the facts at the upper levels of the models. The search for transition functions that allow to prove the target facts on the basis of the initial facts at the upper level. If at some step it was not possible to prove the necessary fact on the basis of the initial facts, the transition to a lower level is made, where a new attempt is made to prove, but already using functions defined at a lower level. Such a descent can be made until the zero level is reached. In case of successful proof of the transition from SM_{t_i} to $SM_{t_{i+k}}$, the performed list of steps is reversed with the exception of

duplicate elements. The resulting path contains links that are established between elements of the static models. Below are the main steps of the algorithm for the synthesis of dynamic models.

1. The search for a path is performed that allows to prove the target fact at the top level of the simulation:

1.1. Among all the input level facts, a search is made for a target fact. If such a fact is found, then control is returned;

1.2. If the target fact is not found, then among the transition functions are those whose output includes the target fact;

1.3. If the function is found, then the verification of its application is performed. The applicability is determined by the conditions of use specified for this function;

1.4. If the conditions of its application are met, the function is added to the current path, and for each input fact of the function, a similar recursive path search is performed.

1.5. If the conditions of application of the function are not satisfied or the function is not found, a similar recursive search for the path to the target fact is performed at all child levels.

1.6. If the path to the target fact at child levels is found, then it is included in the current path. If not found, the search ends. The result of the search is an empty path.

2. If a non-empty path is found that allows to prove the target fact on the basis of the original facts, then the forward move is restored - the order of the functions in the found path is reversed with the exception of duplicate steps.

A fragment of pseudocode implementing the algorithm is shown in Fig. 2 below.2. If a non-empty path is found that allows to prove the target fact on the basis of the original facts, then the forward move is restored - the order of the functions in the found path is reversed with the exception of duplicate steps.

A fragment of pseudocode implementing the algorithm is shown in Fig. 2 below.

According to the proposed algorithm the synthesis problem is solved starting from the top level problem. When solving a problem at the K-th level, a rough synthesis from large blocks of functions is realized. In this case, it is not required to strictly prove the possibility of transition from input data to output state parameters. Inconsistencies of the results at this level are taken into account at corrective synthesis within lower levels of the hierarchy. The multilevel approach to synthesis significantly reduces the time complexity of automatic synthesis. The upper bound of time T_H can be defined as $T_H \approx c\sum_{i=1}^{K} m_i^2 \le c(\sum_{i=1}^{K} m_i)^2$, where c is a constant coefficient; m_i - the number of conditions of the problem at the i-th level. Notice, that m_i is significantly less than the total number of conditions on which problems of program synthesis are solved by traditional methods. This estimate is valid when the number of conditions for multilevel and single-level synthesis problems is the same. Whereas, at each top level, one step of output is equivalent to n_i steps of the lowest level, we can estimate a lower bound for the time T_L of multilevel program synthesis: $T_L \approx c\sum_{i=0}^{K} \frac{m_i^2}{n_i^2} \le c\sum_{i=0}^{K} m_i^2$, where n_i - the number of elements of the i-th level relative to the base level.

```
function findPath(result) {
    localSteps = [];
    if (finalTarget.from.indexOf(result) > -1) {
        return localSteps;
    }

    lastFunction = getFunctionLeadsTo(result);
    if (lastFunction) {
        prevSteps =
lastFunction.args.flatMap(findPath);
        localSteps = prevSteps ++ lastFunction;
    } else {
        isFound = false;
        for (i = 0; i < chidlItems.length; i++) {
            prevSteps = process(chidlItems[i],
result);
            if (prevSteps) {
                localSteps = prevSteps;
                isFound = true;
                break;
            }
        }
    }
    return localSteps;
}

path = findPath(target);
path.forEach((f) => {
    if (resultPath.indexOf(f) === -1) {
        resultPath.push(f);
    } else {
        // feature already available
    }
});
return resultPath;
```

Fig. 2. Pseudocode of mulilevel synthesis algorithm

Most of previous researched of model / program synthesis are built around idea of synthesis of programs / applied structures on top of given formal structures. Our approach is based on idea of dynamic synthesis of formal structures, that can be used later for synthesis of particular algorithms and programs (e.g. scripts).

III. Program system for deductive synthesis

A. General structure

Consider a high-level modular scheme of a software system capable of synthesizing multi-level dynamic models of observed objects. The structure of such a modeling system includes typical subsystems that provide data about objects and interaction with users, as well as a new subsystem - a subsystem of multi-level synthesis. The overall architecture of the deductive modeling system is shown in Figure 3. Data is received and results are output through a certain adapter layer

through established communication channels (for example, TCP/IP networks using arbitrary application protocols HTTP, SOAP, WebSocket; receiving stream media data using the RTSP protocol and etc.) The external systems can be monitoring and control systems.

Below, the main subsystems are discussed in more detail.

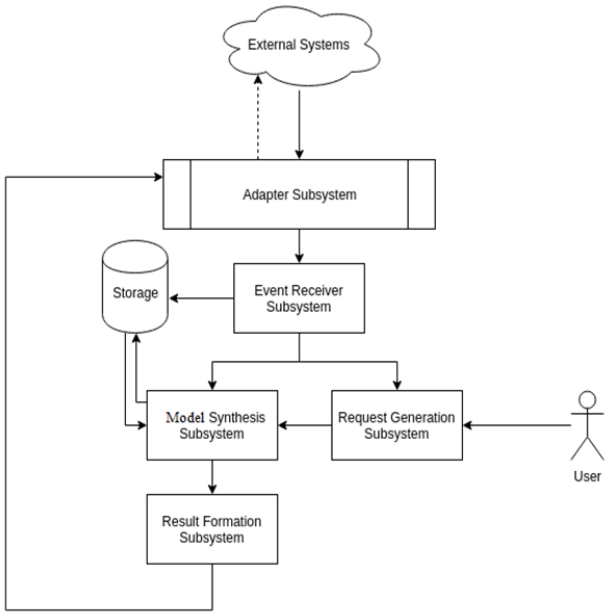

Fig. 3. Multi-level synthesis architecture

The adapter (driver) subsystem provides a communication facade between the modeling system and external systems. At the adapter subsystem level, specific mechanisms are implemented in terms of information acquisition and transfer of results: support of transport and application information protocols (specific, including proprietary), primary analysis and semantization of data, testing communication with external systems, etc. This facade allows unifying flows data entering the modeling system, and also to bring data flows from the modeling system into a specific format used in external systems.

The event receiving subsystem collects all possible information about a simulated object in the form of "events" - atomic units of information characterizing changes in one or another aspect of the functioning of the observed object. These events can carry as fragments of high-level (general) and low-level (private) information about the functioning of the object as a whole and its components. Examples of such events are: "Voltage drop at the power transmission line", "Arming of the premises", "Restoration of communication with the video surveillance subsystem". The event receiving subsystem implements the mechanisms of primary accumulation of such events in the data accumulation subsystem, as well as the function of aggregating information about various elements of the observed object in order to form a higher-level representation of information about the current state of the object. For example, a "Trend Graph" can be built.

The data accumulation subsystem stores all information about the object being modeled, including its state and behavior observed at previous time intervals. (events) and the monitoring process (snapshots of previous states / current state). This information can be placed in a relational / non-relational database.

The query generation subsystem provides automatic or initiative (at the user's command) formation of high-level requests for the synthesis of object models. The subsystem uses not "raw" data about the observed objects, but previously processed by the event receiving subsystem. For tasks of automatic query generation, simplest designs of the "signal-trigger" type can be implemented, or more complex ones based on a tree of logical conditions. Initiative query formation implies the presence of a group of end users who are able to manage the modeling processes, collect clarifying information, enrich the information received from the event receiving subsystem. For example, the validity of calling a response team or determining the composition of measures for the restoration of the video surveillance subsystem can be assessed.

The subsystem of synthesis of models of objects implements a deductive synthesis of models of observable objects. The observed object is described in the formalized language of the synthesis subsystem. The elements of the observed object are lined up in a multilevel hierarchy, in which each level is characterized by a certain possible set of states at a given level, and the state at the upper level depends on the set of states at the lower level. The subsystem defines all possible transitions between states. Any high-level query is presented in the form of a model of the final state of the observed object. Having a model of the current state, a set of transitions between all states and a model of the final state, the subsystem synthesizes a dynamic model of the object. Example of a chain of steps: "CCTV Health-check; check each of the 20 cameras; for each camera, check and compare with the target power control input, tamper input".

The results generation subsystem provides the conversion of simulation results to standard formats or formats that are consistent with external systems. External systems can transmit both all new information about an object obtained as a result of modeling, as well as its individual fragments.

Elements of the considered system are loosely coupled and can implement support for any communication protocols, various deployment and scaling mechanisms. This allows the system to be integrated into existing infrastructures without significant costs.

IV. Case Study

Consider an example of synthesizing dynamic object models for the Internet of Things area. In order to solve complex business problems that are set before IoT [8], it is necessary to provide a reliable technical basis for the operation of these networks. Given the large number of elements that make up the IoT networks, as well as their continuous change, there are a number of problems associated with their monitoring. Monitoring processes can be built and rebuilt based on dynamic models of these networks.

The structure of the simulated IoT network is shown in Fig. 4 [8]. It has three levels, i.e. The main conditions for the

application of the deductive synthesis algorithm are fulfilled. In accordance with the figure, sensors are located at the lower level, translators are at the middle level, the upper level is a generalizing level. In this model, vertical links between all levels are allowed, and horizontal links are only between the elements of the first and second levels.

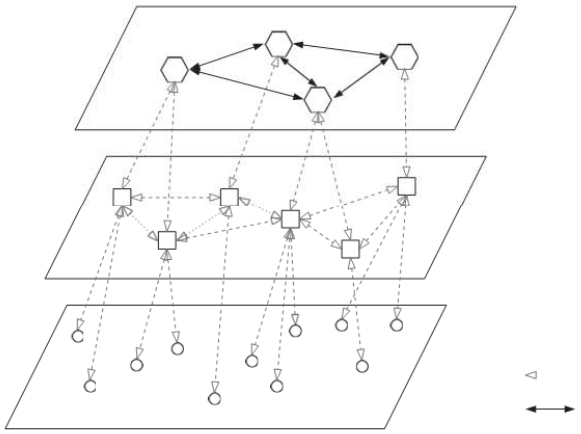

Fig. 4 The structure of the simulated IoT network

As functions defining vertical connections, we will consider the information content of sensors, sensor groups and the network as a whole. Horizontal links show the consistency of data related to different elements of the same level. For example, an increase in temperature should lead to an increase in pressure in the fuel tank. If this dependence is not observed, then two situations are possible: the failure of one of the sensors or the malfunction of the object, the parameters of which are measured.

Consider the synthesis of a dynamic model of the IoT network in the case when the network is fully operational and ensures the collection of all the necessary data. Let the source and target models have the form (Fig.5).

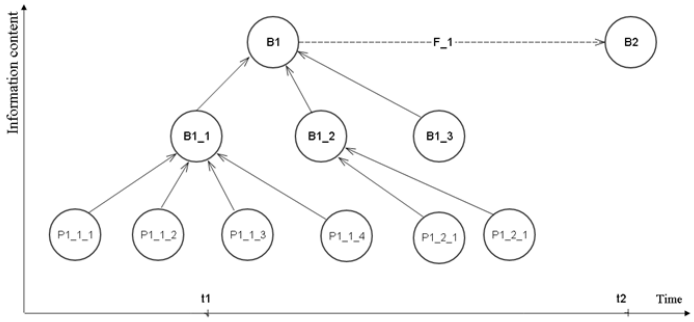

Fig. 5. Model of regular system

With next given facts: *s, w, B1, B2, B1_1, B1_2, B1_3, B2, P1_1_1. P1_1_2. P1_1_3, P1_1_4,* and next given transitions between then (in the formal language of subsystem):

```
[
  {
    "id": "1", "parent": null, "level": 1, "number": 1,
    "functions": [
{ "args": [ "s" ], "result": "B1",  "conditions": "" },
{ "args": [ "B1" ], "result": "B2", "conditions": "pB1==0" },
{ "args": ["B2"   ], "result": "w", "conditions": "pB2==0" }
    ]
  },
  {     "id": "1_1", "parent": "1", "level": 2, "number": 1,
    "functions": [
{ "args": [ "B1" ], "result": "B1_1", "conditions": "" },
{ "args": [ "B1_1" ], "result": "B1_2", "conditions": "pB1_1==0" },
{ "args": [ "B1_2" ], "result": "B1_3", "conditions": "pB1_2==0" },
{ "args": [ "B1_3" ], "result": "B2", "conditions": "pB1_3==0" }
    ]
  },
{ "id": "1_1_1", "parent": "1_1", "level": 3, "number": 1,
    "functions": [{ "args": [ "B1_1" ], "result": "P1_1_1", "conditions": "" },
{ "args": [ "P1_1_1" ], "result": "P1_1_2", "conditions": "pP1_1_1==0" },
{ "args": [ "P1_1_2" ], "result": "P1_1_3", "conditions": "pP1_1_2==0" },
{ "args": [ "P1_1_3" ], "result": "P1_1_4", "conditions": "P1_1_3==0" },
    ]
  },
]
```

, synthesis process may be look like this:

F(s) ->B1; //fact not known
F(B1) ->B1_1 ; //fact not known
F(B1_1)->P1_1_1; ; //determining the amount of information P1_1_1
F(P1_1_1)->P1_1_2 (pP1_1_1==0); //determining the amount of information P1_1_1 + P1_1_2
F(P1_1_2)->P1_1_3 (pP1_1_2==0); //determining the amount of information P1_1_1 + P1_1_2 + P1_1_3
F(P1_1_3)->P1_1_4 (pP1_1_3==0); // determining the amount of information P1_1_1 + P1_1_2 + P1_1_3+ P1_1_4
F(P1_1_4)->B1_2 (pP1_1_4==0); // determining the amount of information B1_1+ B1_2
F(B1_2) ->B1_3 (pB1_2==0) // determining the amount of information B1_1+ B1_2 + B1_3
F(B1_3) ->B2 (pB1_3==0) // comparison of information in B1 andB2
*F(B2) ->**w** (pB2==0) //the amount of information has not significantly decreased*

In case of failure of some data acquisition systems (fig. 6), the overall amount of data could not be retrieved.

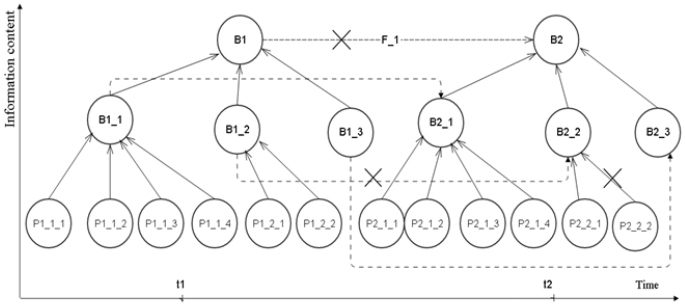

Fig. 6 Model in case of failure of some data acquisition systems

Synthesis process in this case may be look like this:

F(s) ->B1;
F(B1) ->B2 (pB1==0); // comparison of information in B1 andB2
F(B2) ->B2_1; //the amount of information has decreased significantly
F(B2_1) ->B2_2 (pB_2_1); // determining changes in the amount of information B1_1 uB2_1; the amount of information has decreased significantly
F(B2_2) ->P2_2_1; // determining changes in the amount of information B1_2 uB2_2; the amount has decreased significantly
F(P2_2_1)->P2_2_2 (pP2_2_1==0); // on the elementP2_2_2 there's no data Next proof is unavailable

Given example show very simple case of dynamic synthesis of IoT system models, that allows to achieve automatic or semi-automatic monitoring and correction of running processes.

This approach can be easily scaled to much larger systems and appears to be loosely coupled, allowing to be integrated in existing systems seamlessly: it is distributed by design, and different subsystems could be implemented on separated computing platforms and different form-factors: from single cloud / microservice systems to fog-like clusters in LAN.

V. Conclusion

Dynamic synthesis of object models allows to design very complex systems with built-in feedback mechanism and self-control abilities. The developed systems for the synthesis of dynamic models were tested in two areas - the field of IoT and the field of medicine. In both cases, the new solutions were in demand, and the results showed their practical value. In the future, it is planned to conduct testing in a number of other subject areas, in particular, in the field of education, urban economy.

References

[1] J. A. Robinson. A machine – oriented logic based on resolution principle, Journal of the ACM. 12 (1965), pp. 23 – 41.

[2] C. Chang, R. Lee. Symbolic Logic and Mechanical Theorem Proving, New York: Academic, 1973.

[3] S. Yu. Maslov. Teoriadeduktivnykh system ieeprimeneniya (Theory of Deductive Systems and Its Applications), Moscow: Radio I Svyaz', 1986.

[4] E. Kh. Tyugu, M. Ya. Kharf. Algorithms for structural synthesis of programs, Programmirovanie. 4 (1980), pp. 3 – 13.

[5] G. Giacomo, F. Patrizi, S. Sardina. Automatic behavior composition synthesis, Artificial Intelligence. 196 (March 2013), pp. 106 -142.

[6] Osipov V., Lushnov M., Stankova E., Vodyaho "A Inductive Synthesis of the Models of Biological Systems According to Clinical Trials", International Conference on Computational Science and Its Applications (ICCSA 2017). Lecture Notes in Computer Science, vol10404. Springer, Cham. pp. 103-115.

[7] Osipov V.Y."Automatic synthesis of action programs for intelligent robots". Programming and Computer Software,vol. 42, Issue 3, April 2016, pp. 155-160.

[8] Osipov, V.Y., Vodyaho, A.I., Zhukova, N.A., Glebovsky, P.A. "Multilevel Automatic Synthesis of Behavioral Programs for Smart Devices", Proceedings - 2017 International Conference on Control, Artificial Intelligence, Robotics and Optimization, ICCAIRO 2017, 2018-January, pp. 335-340.

[9] V. Osipov, E. Stankova, A Vodyaho, B. Zeno "Finding motifs in Medical Data", 17th International Conference, Computational Science and Its Applications (ICCSA 2017) /Trieste, Italy, July 3-6. Lecture Notes in Computer Science, Part 5. vol 10408. Springer, Cham. pp. 371-386.

[10] Jun Huang and Kun Hua, Managing the Internet of Things. Architectures, Theories and Applications. The Institution of Engineering and Technology, 2016.

[11] Cisco Research: IoT Value: Challenges, Breakthroughs, and Best Practices, Web: https://www.slideshare.net/CiscoBusinessInsights/journey-to-iot-value-76163389

[12] Kurapeev DI, Lushnov MS, OsipovVYu, Vodyaho AI and ZhukovaNA.Synthesis of Integral Models of System Dynamics of an Acid-Base State (ABS) of Patients at Operative Measures // Acta Scientific Volume 3 Issue 3 – 2019, 16-29.2019https://actascientific.com/ASMS-Article-Inpress.php

[13] BorislavLazarovAPPLICATION OF SOME CYBERNETIC MODELS IN BUILDING INDIVIDUAL EDUCATIONAL TRAJECTORY International Journal "Information Models and Analyses" Vol.2 / 2013, Number 1

[14] Technological Platform for Realization of Students' Individual Educational Trajectories in a Vocational SchoolEvald F. ZeeraandAlexsey V. StreltsovIEJME—MATHEMATICS EDUCATION2016, VOL. 11, NO. 7, 2639-2650

[15] Computing Networksfrom cluster to cloud computingPascale Vicat-Blanc, Sébastien Soudan, RomaricGuillier, Brice Goglin ISTE Ltd and John Wiley & Sons, Inc2011

[16] Jun Huang and Kun Hua - Managing the Internet of Things. Architectures, Theories and Applications-The Institution of Engineering and Technology, 2016